Improving Open Source Face Detection by Combining an Adapted Cascade Classification Pipeline and Active Learning

Steven Puttemans[1], Can Ergün[2] and Toon Goedemé[1]
[1]*KU Leuven, EAVISE Research Group, Jan Pieter De Nayerlaan 5, Sint-Katelijne-Waver, Belgium.*
[2]*Istanbul University, Faculty of Economics, Beyazt, 34452 Fatih/Istanbul, Turkey*
{*steven.puttemans, toon.goedeme*}*@kuleuven.be, can.e992@hotmail.com*

Keywords: Open Source, Face Detection, Cascade Classification, Active Learning

Abstract: Computer vision has almost solved the issue of in the wild face detection, using complex techniques like convolutional neural networks. On the contrary many open source computer vision frameworks like OpenCV have not yet made the switch to these complex techniques and tend to depend on well established algorithms for face detection, like the cascade classification pipeline suggested by Viola and Jones. The accuracy of these basic face detectors on public datasets like FDDB stays rather low, mainly due to the high number of false positive detections. We propose several adaptations to the current existing face detection model training pipeline of OpenCV. We improve the training sample generation and annotation procedure, and apply an active learning strategy. These boost the accuracy of in the wild face detection on the FDDB dataset drastically, closing the gap towards the accuracy gained by CNN-based face detectors. The proposed changes allow us to provide an improved face detection model to OpenCV, achieving a remarkably high precision at an acceptable recall, two critical requirements for further processing pipelines like person identification, etc.

1 INTRODUCTION

Face detection (see Figure 1) is a well studied problem in computer vision, and good solutions are presented in literature. However we notice that open source computer vision frameworks like OpenCV (Bradski et al., 2000), offer face detectors based on existing learning techniques, which are unable to yield high accuracies on the available public datasets. A root cause can be the fact that most of these models have been created in the earlier ages of computer vision, when academic research was still interested in older cascade classifier based techniques, like (Viola and Jones, 2001). Academic research evolved and moved on, discovering more promising techniques like convolutional neural networks and loosing interest in well established and proven-to-work algorithms. This resulted in a well known computer vision library still providing a basic face detector, achieving only average detection results on any given dataset.

On the other side, users from the industry interested in turning these open source computer vision frameworks into working applications, get stuck at improving the existing performance of the face detection techniques. Their internal organizational structure does not allow to put efforts into research that tries to boost the performance of current algorithms. Two of the largest issues when trying to improve these existing techniques, are the availability of large amounts of training data and the achievable accuracy limitation reported by academic research of different detection set-ups, using this basic detection model.

In order to fill the gap we decided to investigate how the current cascade classification pipeline for training a face detector inside OpenCV could be adapted to achieve a higher detection accuracy. We do this by adjusting the face annotations, improving the negative training sample collection and by using an active learning strategy to iteratively add hard positive (positive windows classified as negatives in the previous iteration) and hard negative (negative windows classified as positives in the previous iteration) samples to the object detector training process.

Furthermore, we experience that industrial applications of face detection tend to fail due to false positive detections, as seen in Figure 1, because post-detection processing steps depend on a face being available. In the case of a face recognition application, the face detection can be the basis of gathering

Figure 1: Example of *CascadeClassifier.detectMultiScale()* in *OpenCV3.1* framework (*OpenCVBaseline* model).

training and test annotations (Learned-Miller et al., 2016; Wolf et al., 2011). Therefore we aim at improving the available face detection model of OpenCV3.1, based on local binary patterns (Liao et al., 2007), aiming for a very high precision at an acceptable recall.

The remainder of this paper is structured as follows. Section 2 presents related research, while section 3 discusses the used framework and datasets. This is followed by section 4 discussing the proposed approach in detail. Finally section 5 elaborates on the obtained results while section 6 and section 7 sum up conclusions and possible future improvements.

2 RELATED WORK

The OpenCV framework is an open source computer vision framework providing a collection of techniques ranging from basic image segmentation to complex 3D model generation. It steadily grows in size by contributions from a community of both academic researchers and industrial partners, adding recent advances in the computer vision community, while trying to maintain the quality of the existing back-end. We notice that once new functionality is integrated for a longer period of time and heavily used by the community, investments in improving the functionality tends to stop. This could be explained by the fact that the computer vision community has no interest in actual relevant industrial implementations, but rather in pushing the state-of-the-art even further.

Recent advances in computer vision solve face detection by using complex techniques like multi-task cascaded convolutional neural networks (Zhang et al., 2016), convolutional neural networks combined with 3D information (Li et al., 2016) or recurrent convolutional neural networks (Jiang and Learned-Miller, 2016). These techniques yield very promising results, but tend to be fairly complex to implement in actual applications. There is still a lack in well documented and supported open source software libraries that are easy to use. Furthermore we noticed OpenCV is paving the way of integrating these newer techniques, but up till now, their performance inside the OpenCV framework is still not as bug and error free as desired by industrial companies.

The work of Viola and Jones (Viola and Jones, 2001) on face detection using a boosted cascade of weak classifiers has been around for quite some time. It is the standard frontal face detector for many industrial applications so far, like e.g. digital photo cameras. A downside is that many companies use the available software to train their own more complex face detection models, without sharing the models back with the community. This is mainly due to the fact that OpenCV operates under a BSD license, allowing companies to use the code without sharing back any critical adaptations or changes. With our work we aim at improving the currently available frontal face model based on local binary patterns (used as a baseline in this publication) and achieve a model that is able to accurately detect frontal faces in a large variety of set-ups.

One could argue that working on such an old technique is basically a waste of time invested. However, several recent research papers like (Zheng et al., 2016; Puttemans et al., 2016a; Frejlichowski et al., 2016; Puttemans et al., 2016b; Shaikh et al., 2016) prove the importance of such well established techniques for specific cases of industrial object detection.

3 FRAMEWORK AND DATASET

For building our approach we depend on the OpenCV3.1 framework [1], provided and maintained by Intel. We focus on using the *CascadeClassifier* object detection functionality in the C++ interface together with the *opencv_traincascade* application, containing all functionality for building a boosted cascade of weak classifiers using the approach suggested by (Viola and Jones, 2001).

Since the training data of the current OpenCV face detection models is no longer available, we collected a set of face images for training our own frontal face detection model. The images are collected from various sources like YouTube videos and by using a bulk image grabber on social media, imageboards and google image search results. Remark that all of these images are not accompanied by ground truth face labels. On top of that, we created a multi-threaded tool that can use an existing face detection model to efficiently search for valuable face data in a given video, that can then again be added to the training data sets as hard positive and hard negative samples.

For training our new models, we manually annotate 1.000 face regions as positive training windows and combine this with 750.000 negative training windows, automatically grabbed from large resolution negative images not containing faces. As show in Ta-

[1] `http://www.opencv.org`

Table 1: Training data overview for trained models.

Model	#pos	#neg	#stages	#stumps
OpenCVB	xxx	xxx	20	139
BoostedB	1.000	750k	26	137
IterHardPos	1.250	750k	19	146
IterHardPos+	1.500	750k	19	149

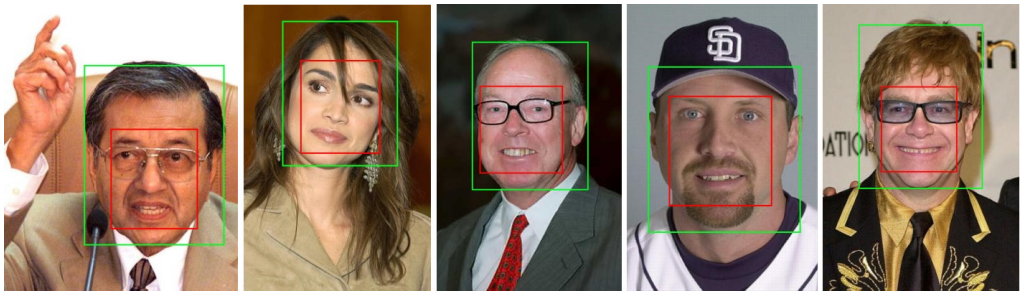


Figure 2: Changing the annotations from full-face to inner-face: (green) OpenCV (red) ours.

ble 1 we then increase the positives dataset for each new iteration with 250 extra hard positive samples. These are gathered from a large set of positive images, in which we know faces occur. Whenever the initial detector is not able to find a face region, a manual intervention is required, asking for a face label, and adding it as a training sample for the following training iteration. The positive training set used for training our final *IterativeHardPositives+* model, can be requested by contacting one of the paper authors.

For validating our new models and comparing them to the existing OpenCV baseline, we use the Face Detection Data Set and Benchmark[2] (FDDB) dataset (Jain and Learned-Miller, 2010). This dataset contains 5171 face annotations in 2845 images collected from the larger Faces in the Wild dataset (Berg et al., 2005). The dataset focuses on pushing the limits in unconstrained face detection. In order to be able to obtain a decent baseline, we converted the existing image annotations into the OpenCV used format, and made them publicly available [3].

4 SUGGESTED APPROACH

In the following subsections we will discuss the different adaptations made to the existing cascade classifier training pipeline, leading to an overall increase in performance, as discussed in section 5.

4.1 Changing the face's region of interest during annotation

When taking a closer look at the output of the OpenCV LBP frontal face detector, we notice that in many cases the detection output contains the complete head, including ears, hair and sometimes even background information. This is due to the OpenCV training data annotations. Figure 2 indicates that OpenCV aimed to include as much facial information as possible to feed to the training algorithm. Since a face detector needs to be generic, we focus on the face part containing the most general features over any given face dataset. In order to reduce the amount of non-trivial face information, we decided to annotate faces as the inner face region only, seen as the red annotations in Figure 1, and as previously suggested by (Mathias et al., 2014) for similar face detection techniques. This approach has several benefits. It removes tons of features from the feature pool of the boosting algorithm, reducing the amount of features that need to be evaluated during model training. Furthermore the inner face is more robust to rotation (both in-plane as out-of-plane). We elaborate more on these in-plane and out-of-plane rotations in section 5.4.

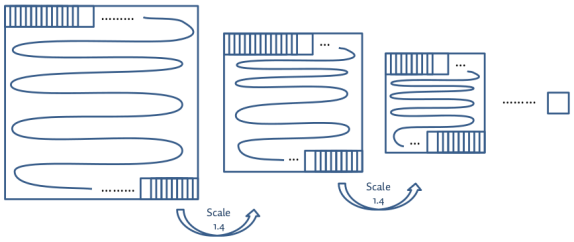
4.2 Adapting the negative training sample collection

OpenCV offers an automated way of collecting negative samples from a set of random background images not containing the object. The algorithm rescales the given negative images to different sizes and uses a sliding window based sequential collecting of negative windows, without any overlap between subsequential windows. Once the set of negative images is completely processed, the process is repeated and adding a pixel offset in each image, to obtain slightly different samples (at pixel level). If a set is traversed multiple times, increasing the offset each time, this process equals applying a pixel shifting sliding window approach, as illustrated in Figure 3(a). While the basic idea of capturing slight differences in your data might be a good starting point, this approach generates a huge amount of negative samples which do not add extra meaningful knowledge to the process, and can thus not be seen as unique samples.

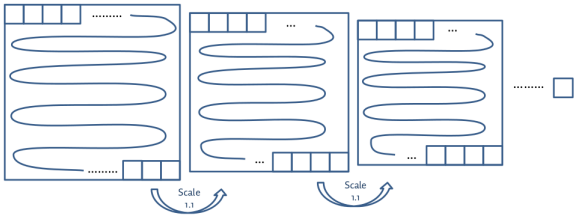
Looking at the boosting process used to train the cascade classifier (by default AdaBoost (Freund et al., 1999)), we notice that each new negative window can only be allowed as negative training sample for

[2] http://vis-www.cs.umass.edu/fddb/

[3] http://eavise.be/OpenSourceFaceDetection/



(a) Original proces inside OpenCV framework.



(b) Suggested adapted pipeline.

Figure 3: Adaptations to the negative windows collection process.

a new stage, if the previous stages do not reject it. If there is only a slight pixel shift for different negatives, then this rejection phase will just evaluate a lot of windows, of which we already know that they will be rejected. Therefore we adapted the interface and removed the pixel offset procedure. By removing this procedure and having no overlap between subsequent negative windows, we introduce a possible loss of valuable information shared around the borders of subsequent samples. This lost information might contain critical knowledge for building a robust detector. To reduce this loss of information we refine the scale generation in the image pyramid. Where OpenCV generates an image pyramid with a scale parameter of 1.4, we decide to lower this scale parameter value 1.1 to ensure that negative samples gathered on different pyramid scales are diverse enough while keeping as much valuable information as possible. This is illustrated in Figure 3(b). By doing so, lost information on sample borders on one scale will be captured by either the previous or the subsequent scale. An extra benefit of refining the scale pyramid, is that the resulting object detection model is more robust to scale changes of the object, able to capture smaller variations in size.

Based on these adaptations it is quite straightforward to collect a large set of negative data samples, something necessary to create a robust face detection model for in the wild applications. Considering a high resolution image of 1.080×1.920 pixels, we can already collect 30.000 negative training samples. This allows us to increase the number of negative samples per stage in our trained cascade classifier to multiple hundred thousands of samples, trying to model the background as good as possible. This will increase training time per stage, but will reduce the amount of stages, and thus make the model faster, less complex and more accurate at detection time.

4.3 Iterative active learning strategy for hard training samples

Supplying heaps of data to machine learning algorithms allow to learn very complex object detection models. The downside is that both in gathering positive and negative training data, it is very difficult to tell which new sample will actually improve the efficiency of the detection model. In order to decide which samples are actually valuable to be added to the process, we apply a technique called active learning. The idea is to use the model trained by the previous iteration and use that model to tell us which samples are valuable (close to the decision boundary) and which are not (no ambiguity in labelling), when adding them to the next iteration training process, as seen in Figure 4. We make a distinction between hard negatives and hard positives as explained below. Furthermore the advantage of active learning is that we limit the amount of manual labour drastically, since we only need to provide labels to new training samples that add extra knowledge to the trained classifier.

4.3.1 Hard negative samples

Hard negative samples are gathered by collecting a set of negative images and running our previously trained face detector on them. All detections returned are in fact negative windows that still trigger a detection, and are thus not assigned to the background yet by our current model. Basically these samples contain information that was not yet captured by the previously collected set of negative samples and thus provide valuable information to the training process.

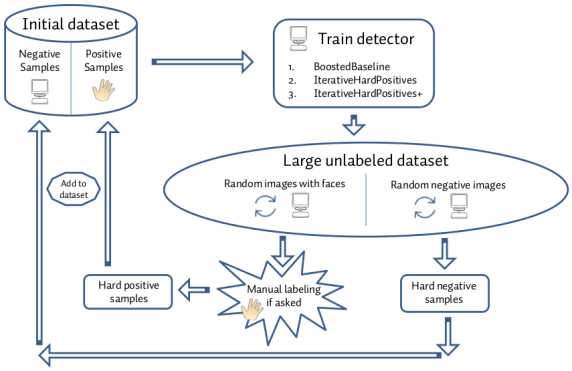


Figure 4: A schematic overview of the active learning process (hand symbol) manual intervention/annotation (computer symbol) fully automated processing.

4.3.2 Hard positive samples

Hard positive samples are gathered by collecting a large set of unlabelled images containing faces. We only know the images contain one (or more) faces, but we do not have a labelled location. On these images, the current face detector is executed (with a low detection certainty threshold) and a piece of software keeps track of images that do not trigger a detection. In that case, an operator is asked to manually select the face region for those triggered images and thus provide labels. This region is stored as a hard positive sample that can still give the model learning interface enough extra valuable knowledge on how it should be learning its model.

4.4 Halting training when negative dataset is consumed

The original OpenCV implementation use pixel-wise offsets in the negative sample grabbing to avoid the training to halt when the original provided dataset is completely consumed in a first run. In section 4.2 we already describe that using these pixel shifted windows is overkill and adds a lot of redundant data. We halt the training when the negative dataset is completely consumed. Once that happens we give the operator two possibilities. Either we allow to add extra images to the negative image dataset, or we return the amount of negative samples that was grabbed in the last stage before the training was halted. This allows the operator to finalize the last stage with this exact amount of samples and thus train a model using every single negative sample window, completely consuming the available negative dataset.

4.5 Using the adaptations to train different face detection models

By smartly combining all the adaptations suggested in section 4.1, 4.2 and 4.3 we train different face detection models where we iteratively try to improve the accuracy of the obtained model. Table 1 describes the training data used for these models, in combination with the number of model stages and the number of features (each forming a stump/binary decision tree) selected by the boosting process.

Our first model (referred to as *'BoostedBaseline'*) can be seen as our baseline we iteratively try to improve by applying the active learning strategy. We limit the training to only incorporate stumps, which are single layer decision trees. One might argue that using more complex decision trees is more profitable but previous research shows that using more complex trees actually slows the detection process (Reyzin and Schapire, 2006), because more features need to be evaluated in early stages. For each boosted learning model, the increase in performance when adding features should outweigh the complexity and thus the processing time. Our model training is halted when the collected set of random negative background images is completely consumed.

For the second model, referred to as *'IterativeHardPositives'*, we add 250 hard positive training samples collected through the active learning procedure, trying to improve the recall rate of the detector. We also gather a limited set of hard negatives and add those to the training set. We noticed that adding these extra quality samples pushes the recall rate while slightly increasing the precision rate. The third and final model, referred to as *'IterativeHardPositives+'*, is again improved by providing 250 extra hard positive samples, in an attempt to push the reported recall even further.

5 RESULTS

5.1 Performance of trained models

Figure 5 compares the trained models (*BoostedBaseline*, *IterativeHardPositives* and *IterativeHardPositives+*) from section 4.5 to the *OpenCVBaseline* detector on the FDDB test dataset. Performance is measured using precision-recall plots. We notice a generally large improvement of our self trained models (green, red and blue curve) over the OpenCV baseline (black curve). The OpenCV baseline model is only able to achieve a recall of about 40% (meaning 4 out of 10 objects are detected) at a precision of 40% (of all the detections returned, only 4 out of 10 are actual objects) for its optimal point. Of course one can make a trade-off and decide to sacrifice recall for a higher precision. Nonetheless the current OpenCV model is not able to detect objects with a certainty higher than 50% on the given FDDB dataset, containing a wild variety of faces in very challenging conditions.

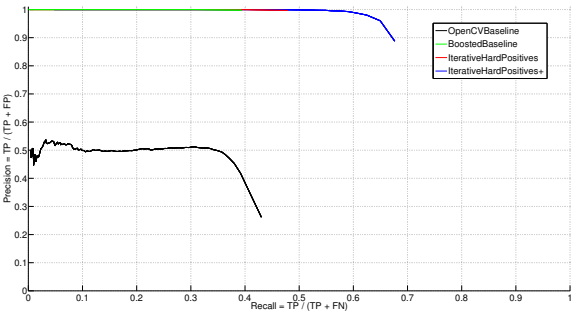


Figure 5: Precision-Recall for all models on FDDB dataset.

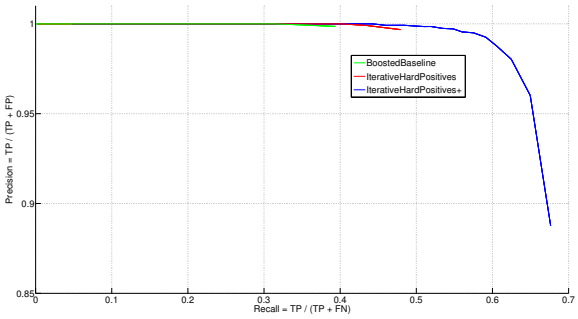


Figure 6: Close-up of PR curves of our detection models.

Compared to the *OpenCVBaseline* detector, at the optimal recall of 40% for that model, our *BoostedBaseline* detector already increases the precision towards 99.5%, almost completely removing the existence of false positive detections. Furthermore, each of our subsequent models, as seen in the close-up in Figure 6, increases the recall further without sacrificing the very high precision rate. At a recall value of 60%, a 50% increase compared to the *OpenCVBaseline* detector, our *IterativeHardPositives+* detector only has a slight drop to 99% precision. As an optimal working point our *IterativeHardPositives+* model reaches a precision of 90% at a recall of 68%.

While many papers on face detection use precision recall curves to compare detection models efficiently, the official FDDB evaluation criteria is based on the true positive rate compared to the number of false positive detections. We include this comparison for both the *OpenCVBaseline* detector and our *IterativeHardPositives+* detector, as seen in Figure 7. We also compare our technique to some state-of-the-art face detection algorithms based on neural networks like FastR-CNN (Jiang and Learned-Miller, 2016), ConvNet3D (Li et al., 2016) and MultiTaskCNN (Zhang et al., 2016). This clearly shows that we already close the gap between cascade classifiers and neural networks a lot, while still having room for improvement.

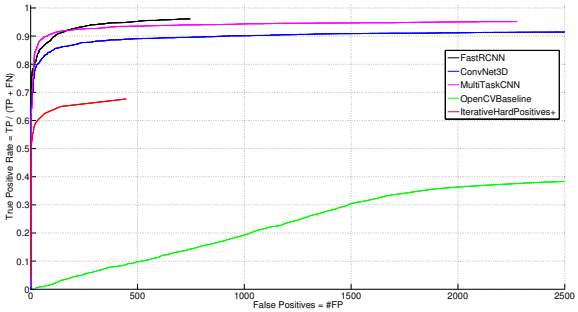


Figure 7: Evaluation for FDDB dataset, comparing our algorithm to neural network based approaches.

Table 2: Timing results comparing both OpenCV baseline and self trained models for the FDDB dataset.

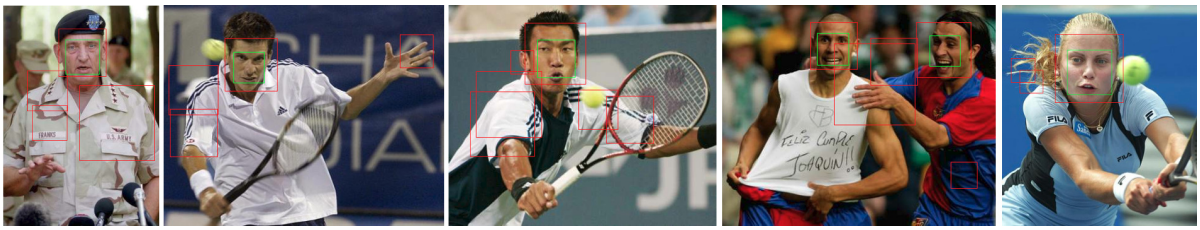
Model	Whole Set	Per Image
OpenCVBaseline	9 min 30 sec	0.20 sec
BoostedBaseline	6 min 8 sec	0.13 sec
IterativeHardPos	7 min 7 sec	0.15 sec
IterativeHardPos+	9 min 6 sec	0.19 sec

5.2 Influence of adaptations to processing time

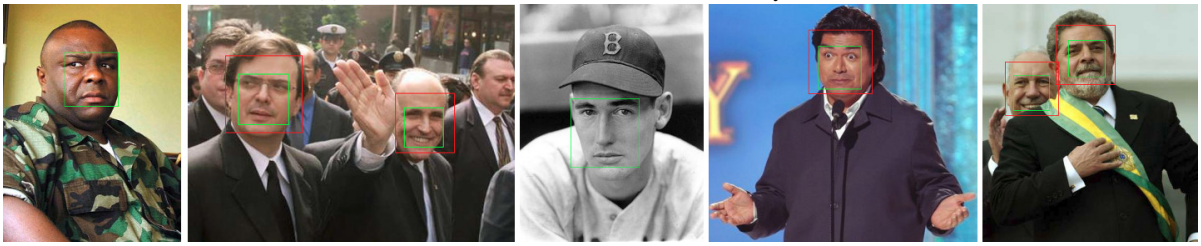
One must make sure that adding all this extra training data does not make the model overly complex and slow during detection time. As shown in Table 1 we have only a limited increase in used features as stump classifiers, while adding 50% more valuable positive training data. Furthermore the complexity in number of stages drops with our models. Since processing time is a key feature for many computer vision approaches applied in embedded systems, we took the liberty of measuring processing time over the complete FDDB test set, which can be seen in Table 2. We average the timings to receive a timing per image, given the average resolution of the test images is 400×300 pixels. These timings are performed on a Intel(R) Xeon(R) CPU E5-2630 v2 system set-up. Our OpenCV build is optimized using the Threading Building Blocks for parallel processing. We clearly see, although we are using more features in our model, that the processing time of our *IterativeHardPositives+* model does not exceed the processing time of the *OpenCVBaseline* model. Furthermore, if we use our *BoostedBaseline* or *IterativeHardPositives* detector, we process images remarkably faster than the *OpenCVBaseline* detector.

5.3 A visual confirmation

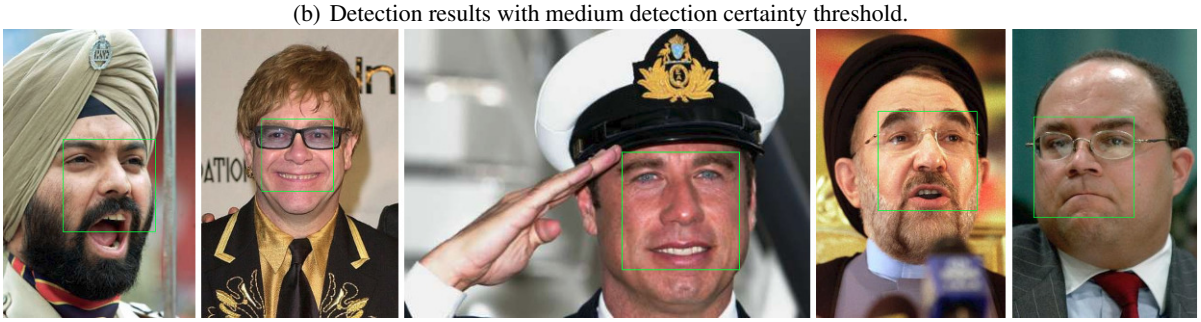
Figure 8 shows some visual detection output of our algorithm. We start by selecting a low detection certainty threshold (Figure 8(a)) which clearly shows that both models are able to find faces, but immediately shows the downside of the OpenCV model, which generates a lot of false positive detections. We increase the detection certainty threshold to a mediate level (Figure 8(b)) and notice that both OpenCV and our own trained model are able to find faces, but gradually OpenCV starts to miss faces that are still detected by our model. Finally when setting a high detection certainty threshold (Figure 8(c)), we see that OpenCV misses a lot of faces that are still found by our model. But even in the case that our model detects



(a) Detection results with low detection certainty threshold.



(b) Detection results with medium detection certainty threshold.



(c) Detection results with high detection certainty threshold.

(d) Cases where both detectors fail (high certainty threshold) or where OpenCV finds a detection while we do not.

Figure 8: Detection results and failures on FDDB dataset for (red) *OpenCVBaseline* and (green) *IterHardPos+* model.

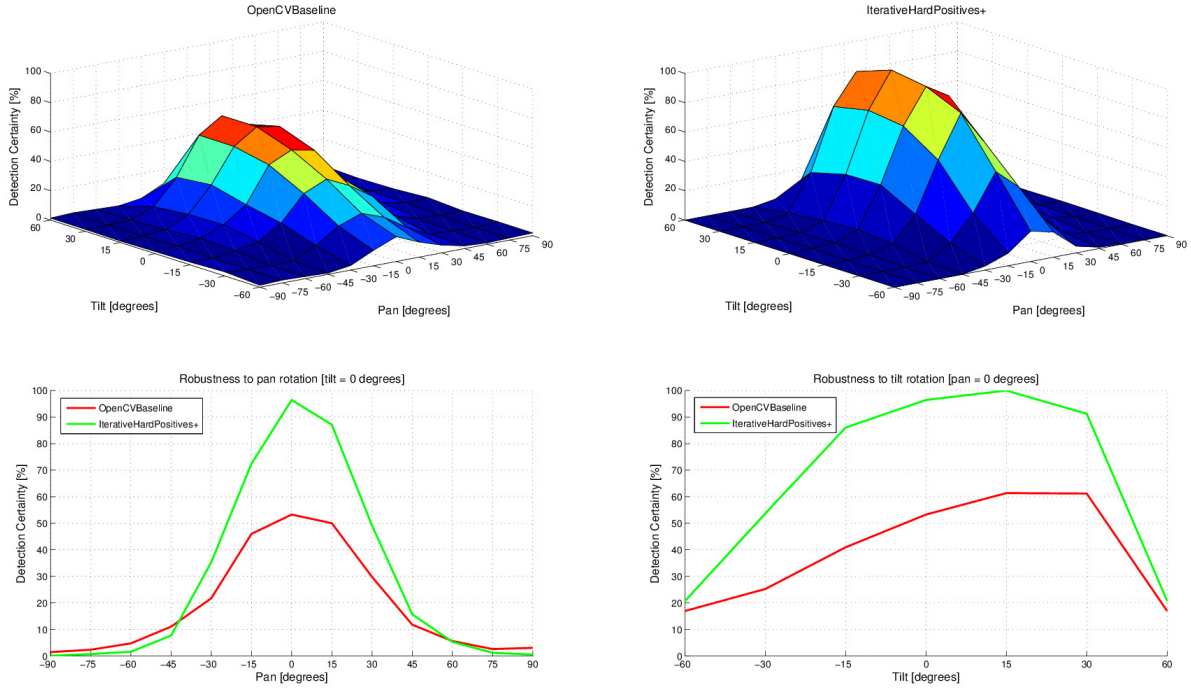


Figure 9: Testing out-of-plane rotational robustness for both *OpenCVBaseline* and the *IterativeHardPositives+* detector.

more faces than OpenCV we still find cases where both models fail or where OpenCV actually finds a face that our models does not capture, as seen in Figure 8(d). These undetected faces could be used as hard positive training samples but then we would need to search for a new database for evaluation purposes in order not to introduce dataset bias.

5.4 Testing out-of-plane rotation robustness

As already stated in section 4.1 reducing the annotation region, which directly influences the face region that the detector will return, helps improving the out-of-plane rotation of the face detector. To test this, we evaluated the *OpenCVBaseline* and the *IterativeHardPositives+* detector on the Head Pose Image Database (Gourier et al., 2004), as seen in Figure 9. This dataset contains a set of 30 sequences (15 persons, 2 sequences per person) where people sequentially look at different positions, each associated with a pan (in the range [-90°,+90°]) and a tilt angle (in the range [-60°,60°]). At each position, we execute both detectors and return the detection certainty of the models. averaged over the 30 sequences. We use the highest returned detection score on the dataset as the outer bound of our score range and normalize all other values for this maximum. We see that in both pan and tilt angle evaluations our *IterativeHardPositives+* detector clearly outperforms the *OpenCVBaseline* detector. Especially in the tilt angle range, we see a large increase in efficiency. This extra test also confirms that at a full frontal face, the *IterativeHardPositives+* detector has about double the detection certainty as the *OpenCVBaseline* detector, which was already clearly noticeable in Figure 5.

6 CONCLUSIONS

The goal of this paper is to suggest adaptations to the current existing cascade classification pipeline in the open source computer vision framework OpenCV with the eye on improving its frontal face detection model. We aim at reducing the huge amount of false positive detections, by guaranteeing a high precision, while maintaining the recall as high as possible, to detect as many faces as possible. We test our approach on the publicly available FDDB face dataset and prove that our adaptations to the pipeline generate an enormous increase in performance. Using our *IterativeHardPositives+* detector, we achieve an increase in recall to 68% while maintaining a high precision of 90%. Compared to a 40% precision at 40% recall for the current implementation, this is quite impressive.

The suggested adaptations to the framework and the model clearly have benefits over the currently available model. Imagine a case where the output of the face detector is used to perform face recog-

nition. In such cases we aim at a precision that is as high as possible, since we want to ensure that the pipeline following on the actual detection, is not provided with rubbish but with an actual face. Furthermore our model is able to find more faces in the wild and is more robust to out-of-plane rotations compared to the OpenCV baseline model.

We should take into account that we will never hit a 100% recall on datasets like FDDB, due to some high out-of-plane rotations, as seen in Figure 8(d). However one could argue that faces with an out-of-plane rotation of more than 45 degrees should be found by a profile face detector and combine both detectors together, as suggested in (Hulens et al., 2016).

7 FUTURE WORK

As future work we suggest to push the accuracy of the face detection model in the OpenCV framework even further. We have still room to increase the amount of hard positives samples, aiming for an even higher recall rate. A good start could be to run our *IterativeHardPositives+* detector on the FDDB dataset and use the returned hard positive faces as training data. However this will force us to look at new evaluation datasets besides FDDB to avoid dataset bias.

At the moment the model is only evaluated on a single in-plane rotation. Like suggested in (Puttemans et al., 2016a) we could build a rotational 3D matrix of the image and apply our *IterativeHardPositives+* detector several times to incorporate these in-plane rotations. This would allow us to find more faces and push the performance of our pipeline even further.

ACKNOWLEDGEMENTS

This work is supported by the KU Leuven, Campus De Nayer and the Flanders Innovation & Entrepeneurship (AIO).

REFERENCES

Berg, T. L., Berg, A. C., Edwards, J., and Forsyth, D. (2005). Whos in the picture. *Advances in neural information processing systems*, 17:137–144.

Bradski, G. et al. (2000). The opencv library. *Doctor Dobbs Journal*, 25(11):120–126.

Frejlichowski, D., Gościewska, K., Forczmański, P., Nowosielski, A., and Hofman, R. (2016). Applying image features and adaboost classification for vehicle detection in the sm4public system. In *Image Processing and Communications Challenges 7*, pages 81–88.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Gourier, N., Hall, D., and Crowley, J. L. (2004). Estimating face orientation from robust detection of salient facial features. In *ICPR International Workshop on Visual Observation of Deictic Gestures*.

Hulens, D., Van Beeck, K., and Goedemé, T. (2016). Fast and accurate face orientation measurement in low-resolution images on embedded hardware. In *Proceedings of VISIGRAPP*, volume 4, pages 538–544.

Jain, V. and Learned-Miller, E. (2010). Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst.

Jiang, H. and Learned-Miller, E. (2016). Face detection with the faster r-cnn. *arXiv preprint arXiv:1606.03473*.

Learned-Miller, E., Huang, G. B., RoyChowdhury, A., Li, H., and Hua, G. (2016). Labeled faces in the wild: A survey. In *Advances in Face Detection and Facial Image Analysis*, pages 189–248.

Li, Y., Sun, B., Wu, T., Wang, Y., and Gao, W. (2016). Face detection with end-to-end integration of a convnet and a 3d model. *arXiv preprint arXiv:1606.00850*.

Liao, S., Zhu, X., et al. (2007). Learning multi-scale block local binary patterns for face recognition. In *Advances in Biometrics*, pages 828–837.

Mathias, M., Benenson, R., Pedersoli, M., and Van Gool, L. (2014). Face detection without bells and whistles. In *European Conference on Computer Vision*, pages 720–735.

Puttemans, S., Van Ranst, W., and Goedemé, T. (2016a). Detection of photovoltaic installations in rgb aerial imaging: a comparative study. In *GEOBIA2016*.

Puttemans, S., Vanbrabant, Y., Tits, L., and Goedemé, T. (2016b). Automated visual fruit detection for harvest estimation and robotic harvesting. In *IPTA2016*.

Reyzin, L. and Schapire, R. E. (2006). How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd international conference on Machine learning*, pages 753–760.

Shaikh, F., Sharma, A., Gupta, P., and Khan, D. (2016). A driver drowsiness detection system using cascaded adaboost. *Imperial Journal of Interdisciplinary Research*, 2(5).

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages I–511.

Wolf, L., Hassner, T., and Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *CVPR*, pages 529–534.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. *arXiv preprint arXiv:1604.02878*.

Zheng, Y., Yang, C., Merkulov, A., and Bandari, M. (2016). Early breast cancer detection with digital mammograms using haar-like features and adaboost algorithm. In *SPIE Commercial+ Scientific Sensing and Imaging*, pages 98710D–98710D. International Society for Optics and Photonics.